

# Delete a Node

 locked

Problem

Submissions

Leaderboard

Discussions

This challenge is part of a tutorial track by [MyCodeSchool](#) and is accompanied by a video lesson.

Delete the node at a given position in a linked list and return a reference to the head node. The head is at position 0. The list may be empty after you delete the node. In that case, return a null value.

## Example

***l*list** = 0 → 1 → 2 → 3  
***position*** = 2

After removing the node at position 2, ***l*list'** = 0 → 1 → 3.

## Function Description

Complete the *deleteNode* function in the editor below.

*deleteNode* has the following parameters:

- *SinglyLinkedListNode pointer llist*: a reference to the head node in the list
- *int position*: the position of the node to remove

## Returns

- *SinglyLinkedListNode pointer*: a reference to the head of the modified list

## Input Format

The first line of input contains an integer ***n***, the number of elements in the linked list.  
Each of the next ***n*** lines contains an integer, the node data values in order.  
The last line contains an integer, ***position***, the position of the node to delete.

## Constraints

- $1 \leq n \leq 1000$
- $1 \leq \text{list}[i] \leq 1000$ , where ***list***[*i*] is the *i*<sup>th</sup> element of the linked list.

## Sample Input

```
8
20
6
2
19
7
4
15
9
3
```

## Sample Output

```
20 6 2 7 4 15 9
```

## Explanation

The original list is **20 → 6 → 2 → 19 → 7 → 4 → 15 → 9**. After deleting the node at position **3**, the list is **20 → 6 → 2 → 7 → 4 → 15 → 9**.



Submissions: 120

Max Score: 40

Difficulty: Easy

Rate This Challenge:



[More](#)

C++14



```
1 ▶ #include <iostream>
2
3 using namespace std;
4
5 class SinglyLinkedListNode {
6 public:
7     int data;
8     SinglyLinkedListNode *next;
9
10    SinglyLinkedListNode(int node_data) {
11        this->data = node_data;
12        this->next = nullptr;
13    }
14 };
15
16 class SinglyLinkedList {
17 public:
18     SinglyLinkedListNode *head;
19     SinglyLinkedListNode *tail;
20
21    SinglyLinkedList() {
22        this->head = nullptr;
23        this->tail = nullptr;
24    }
25
26    void insert_node(int node_data) {
27        SinglyLinkedListNode* node = new SinglyLinkedListNode(node_data);
28
29        if (!this->head) {
30            this->head = node;
31        } else {
32            this->tail->next = node;
33        }
34
35        this->tail = node;
36    }
37 };
38
39 void print_singly_linked_list(SinglyLinkedListNode* node, string sep, ofstream& fout) {
40     while (node) {
41         fout << node->data;
42
43         node = node->next;
44
45         if (node) {
46             fout << sep;
47         }
48     }
49 }
50
51 void free_singly_linked_list(SinglyLinkedListNode* node) {
52     while (node) {
```

```

53     SinglyLinkedListNode* temp = node;
54     node = node->next;
55
56     free(temp);
57 }
58 }
59
60 /*
61  * Complete the 'deleteNode' function below.
62  *
63  * The function is expected to return an INTEGER_SINGLY_LINKED_LIST.
64  * The function accepts following parameters:
65  * 1. INTEGER_SINGLY_LINKED_LIST llist
66  * 2. INTEGER position
67  */
68
69 /*
70  * For your reference:
71  *
72  * SinglyLinkedListNode {
73  *     int data;
74  *     SinglyLinkedListNode* next;
75  * };
76  */
77
78 SinglyLinkedListNode* deleteNode(SinglyLinkedListNode* llist, int position) {
79
80 }
81
82 int main()
83 {
84     ofstream fout(getenv("OUTPUT_PATH"));
85
86     SinglyLinkedList* llist = new SinglyLinkedList();
87
88     int llist_count;
89     cin >> llist_count;
90     cin.ignore(numeric_limits<streamsize>::max(), '\n');
91
92     for (int i = 0; i < llist_count; i++) {
93         int llist_item;
94         cin >> llist_item;
95         cin.ignore(numeric_limits<streamsize>::max(), '\n');
96
97         llist->insert_node(llist_item);
98     }
99
100     int position;
101     cin >> position;
102     cin.ignore(numeric_limits<streamsize>::max(), '\n');
103
104     SinglyLinkedListNode* llist1 = deleteNode(llist->head, position);
105
106     print_singly_linked_list(llist1, " ", fout);
107     fout << "\n";
108
109     free_singly_linked_list(llist1);
110
111     fout.close();
112
113     return 0;
114 }

```

Line: 24 Col: 1

